What is claimed is:

1. A method for transferring a set of files, the method comprising:

receiving metadata and stub files associated with the set of files at a destination fileserver;

updating a location component in the destination fileserver to maintain a list of repository nodes that are associated with each file in the set of files;

replacing each stub file with the full content of the file associated with the stub file; and

while replacing each stub file, upon receipt of a client request for a specified file in the set of files, if the full content of the specified file has not yet been transferred, then replacing the stub file for the specified file with the specified file's full content, wherein replacing the stub file for the specified file is a higher priority task than replacing the stub files for non-requested files.

2. The method of claim 1 wherein the metadata is received at a destination fileserver from a repository node.

3. The method of claim 1 wherein the method further comprises:

prior to receiving metadata, receiving destination fileserver selection data.

4. The method of claim 1 wherein the method further comprises: prior to receiving metadata, receiving source share selection data.

5. The method of claim 1 wherein the set of files is the set of files that have been accessed during a specified period and wherein replacing each stub file comprises recursively replacing the stub file associated

with the file that was most-recently accessed until all the stub files in the set of files have been replaced.

6. The method of claim 5 wherein the specified period is a most-recent period.

7. The method of claim 1 wherein the location component is a location cache.

8. A data protection system comprising:

a fileserver having:

a file system operative to store client files;

a fileserver API operative to communicate with a repository;

a fileserver file transfer module in communication with the file system and operative to receive files for the file system from at least one repository; and

a recovery service in communication with the fileserver API and with the file system and operative to transfer a set of files, the recovery service having:

a receiving component operative to receive metadata and stub files associated with the set of files at the fileserver;

a location updating component in communication with the receiving component and operative to maintain a list of repository nodes that are associated with each file in the set of files; and

a stub file replacement component in communication with the receiving component and operative to replace each stub file with the full content of the file associated with the stub file.

9. The system of claim 8 wherein the system further comprises

a filter driver operative to intercept input/output activity initiated by client file requests and to maintain a list of modified and created files since a prior backup;

a policy cache operative to store a protection policy associated with a share;

a mirror service in communication with the filter driver and with the policy cache, the mirror service operative to prepare modified and created files in a share to be written to a repository as specified in the protection policy associated with the share.

10. The system of claim 9 wherein the system further comprises:

a location cache in communication with the mirror service and operative to indicate which repository should receive an updated version of an existing file; and

a location manager coupled to the location cache and operative to update the location cache when the system writes a new file to a specific repository node.

11. The system of claim 8 wherein the system further comprises

a local repository having:

a local repository node API adapted for communicating with the fileserver API;

a local repository file transfer module in communication with the fileserver file transfer module and adapted for transferring files to the fileserver file transfer module; and

a data mover in communication with the local repository API and operative to supervise the replication of files from the local repository to the fileserver.

12. The system of claim 11 wherein the fileserver API is operative to communicate with a network and wherein the system further comprises:

a remote repository having:

a remote repository node API adapted for communicating with the network;

a remote repository file transfer module in communication with the local file transfer module and adapted for transferring files to the fileserver file transfer module; and

a data mover in communication with the remote repository API and operative to supervise the replication of files from the remote repository to the fileserver.

13. A method for storing data, the method comprising:

providing a fileserver having:

a file system operative to store client files;

a policy component operative to store a protection policy associated with a set of files;

a mirror service in communication with the policy cache, the mirror service operative to prepare modified and created files in a set of files to be written to a repository as specified in the protection policy associated with the set of files;

a fileserver API coupled to the mirror service and operative to communicate with a repository;

a fileserver file transfer module in communication with the file system and operative to transfer files from the file system to at least one repository;

determining a caching level as stored in the policy component; and

recursively, determining a utilization of the fileserver;

comparing the caching level against the utilization; and

if the utilization exceeds the caching level, then

creating a file migration candidate list;

staging out one candidate file;

replacing the candidate file with a stub file; and

determining if the utilization of the fileserver still exceeds the caching level.

14. The method of claim 13 wherein if determining if the utilization of the fileserver still exceeds the caching level indicates that the utilization exceeds the caching level then staging out another candidate file on the candidate list and again determining if the utilization of the fileserver exceeds the caching level.